

(19) World Intellectual Property  
Organization  
International Bureau



(43) International Publication Date  
21 October 2004 (21.10.2004)

PCT

(10) International Publication Number  
**WO 2004/090739 A1**

(51) International Patent Classification<sup>7</sup>: **G06F 15/00**,  
17/30

(74) Agents: SAMUEL, Richard, I. et al.; Goodwin Procter  
LLP, 103 Eisenhower Parkway, Roseland, NJ 07068 (US).

(21) International Application Number:  
PCT/US2004/010027

(81) Designated States (unless otherwise indicated, for every  
kind of national protection available): AE, AG, AL, AM,  
AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN,  
CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI,  
GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE,  
KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD,  
MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG,  
PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM,  
TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM,  
ZW.

(22) International Filing Date: 1 April 2004 (01.04.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:  
60/459,329 1 April 2003 (01.04.2003) US

(71) Applicant (for all designated States except US): PLAN-  
ETII USA INC. [US/US]; Suite 305, 560 South Winches-  
ter Boulevard, San Jose, CA 95128 (US).

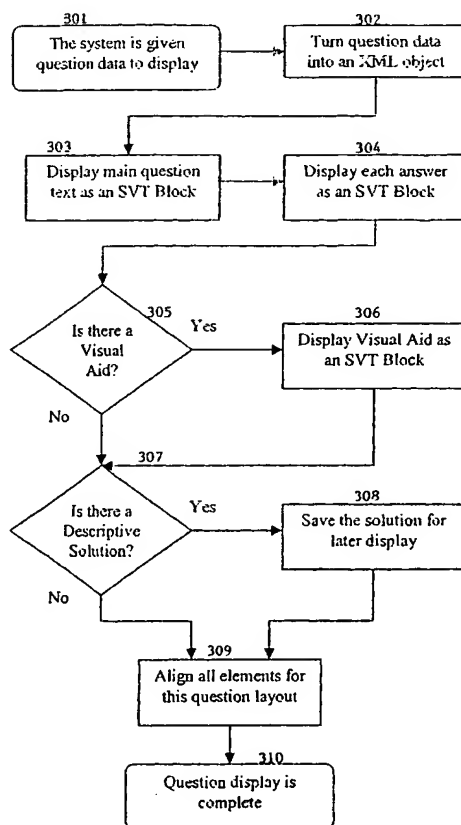
(84) Designated States (unless otherwise indicated, for every  
kind of regional protection available): ARIPO (BW, GH,  
GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),  
Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), Euro-  
pean (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR,  
GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK,

(72) Inventors: FUSION, Joe; Portland, OR 97210-3223  
(US). KYUNG, Lee; San Jose, CA 95126 (US). CHENG,  
Lewis; San Jose, CA 95126 (US).

[Continued on next page]

(54) Title: ADVANCED VERSATILE LAYOUT AND RENDERING SYSTEM, METHOD AND PRODUCT

Question Display Flowchart



(57) Abstract: Provided herein is a system, method and computer program product for effectively arranging and rendering multimedia information using for example, Macromedia's Flash™ technology, the Extensible Markup Language (XML) language and the Mathematical Markup Language (MathML). The invention utilizes question data (301), possible visual aids (305), possible descriptive solutions (307), and element alignment (309).



TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

**Published:**

- with international search report
- before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments

JC09 Rec'd PCT/PTO 30 SEP 2005

## 1 TITLE

2 Advanced Versatile Layout and Rendering System, Method and Product

3

## 4 CLAIM OF PRIORITY/CROSS REFERENCE OF RELATED

## 5 APPLICATION(S)

6 This application claims the benefit of priority of United States Provisional

7 Application Number 60/459,329, filed April 1, 2003, entitled "Advanced

8 Versatile Layout and Rendering System," hereby incorporated in its entirety

9 herein.

10

## 11 STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR

## 12 DEVELOPMENT

13 Not applicable.

14

## 15 COPYRIGHT/TRADEMARK STATEMENT

16 A portion of the disclosure of this patent document may contain material which is

17 subject to copyright and/or trademark protection. The copyright/trademark owner

18 has no objection to the facsimile reproduction by anyone of the patent document

19 or the patent disclosure, as it appears in the Patent Office patent files or records,

20 but otherwise reserves all copyrights and trademarks.

1 BACKGROUND

2 1. Field of the Invention

3 The present invention relates generally to graphical rendering systems and more  
4 particularly to a system, apparatus, process and article of manufacture for  
5 providing improved interactive, graphical applications using, for example,  
6 Macromedia Flash™ technology available from Macromedia Inc., the Extensible  
7 Markup Language (XML) language and the Mathematical Markup Language  
8 (MathML).

9  
10 Details of Macromedia Flash™ technology, including preferred software and  
11 hardware environments, can be found in various sources such as: Macromedia's  
12 website, <<http://www.macromedia.com>>, conference notes from FlashForward  
13 and Macromedia Ucon; in several books published, for example by Friends of Ed,  
14 O'Reilly & Co. and Macromedia; articles and user forums on websites such as  
15 We're Here, FlashKit, UltraShock, and Figleaf's FlashCoders.

16  
17 Details of XML and MathML, including preferred software and hardware  
18 environments, can also be found in various sources including the W3C's website,  
19 <<http://www.w3c.org>>. Specifically, the current MathML specification entitled,

1 "Mathematical Markup Language (MathML) Version 2.0," is located at  
2 <<http://www.w3.org/TR/2001/REC-MathML2-20010221>>.

3

4 Each of the above references and any additional reference provided herein are  
5 incorporated in its entirety herein.

6

## 7 2. Description of Related Art

8

9 Currently, multimedia information comprising text, graphics, mathematical  
10 expressions, symbols and other indicia (collectively, referred to as "mathematical  
11 expressions"), etc., is developed and presented as follows:

12

13 Current web browsers, e.g., Internet Explorer™, Netscape™, etc., display stylized  
14 text and graphics/images however, only a few lesser known web browsers contain  
15 native MathML support, thereby limiting use.

16

17 Another technique involves manually generating the entire multimedia layout  
18 using a graphical creation and rendering program such as Macromedia Flash  
19 MX™ development tool by Macromedia, Inc. However, manual layout is acutely

1    impractical due to high development costs and the severe constraints placed on  
2    maintainability, scalability and portability.

3

4    The Macromedia Flash Player™ and its associated Flash Plug-in™ display  
5    stylized text that is formatted as a Hypertext Markup Language (HTML) object.  
6    However, such an arrangement offers no integration with graphics or  
7    mathematical expressions.

8

9    Existing software products display mathematical expressions by rendering  
10    MathML objects to a graphical format such as the Joint Photographic Experts  
11    Group (JPEG) format. However, to be effective the graphics must be displayed  
12    inline with text. Further pre-rendering the graphics or rendering them  
13    dynamically to address the preceding requirement is costly in performance,  
14    storage and bandwidth, as well as licensing fees.

15

16    The present invention solves the aforementioned limitations of the prior art.  
17    Specifically, the present invention is a comprehensive solution for effectively  
18    arranging and rendering multimedia information comprising mixed data types  
19    including: text, graphics, animations, video and mathematical expressions. The  
20    mixed data may be displayed in various sizes and formats and is in a portable and

1 maintainable format. In addition, there is support for certain technologies, such  
2 as, Macromedia Flash™, multilingual and Unicode support, and client-server n-  
3 tier implementations. Furthermore, performance and costs are kept at acceptable  
4 levels.

5

6 Additional aspects, features and advantages of the present invention will become  
7 better understood with regard to the following description.

8

#### 9 BRIEF DESCRIPTION OF THE DRAWING(S)

10 Referring briefly to the drawings, embodiments of the present invention will be  
11 described with reference to the accompanying drawings in which:

12

13 Figure 1A illustrates an exemplary system constructed in accordance with the  
14 teachings expressed herein.

15

16 Figure 1B illustrates an exemplary networked system constructed in accordance  
17 with the teachings expressed herein.

18

19 Figure 2 illustrates an exemplary data format in accordance with the teachings  
20 expressed herein.

1

2 Figure 3 illustrates an exemplary process flow in accordance with the teachings  
3 expressed herein.

4

5 Figure 4 illustrates an exemplary psuedo-code listing implementing the  
6 embodiment of FIG. 3 in accordance with the teachings expressed herein.

7

8 Figure 5 illustrates an additional and exemplary process flow in accordance with  
9 the teachings expressed herein.

10

11 Figure 6 illustrates an exemplary psuedo-code listing implementing the  
12 embodiment of FIG. 5 in accordance with the teachings expressed herein.

13

14 Figure 7 illustrates an exemplary user interface in accordance with the teachings  
15 expressed herein.

16

## 17 DETAILED DESCRIPTION OF THE PRESENT INVENTION

18 Referring more specifically to the drawings, for illustrative purposes the present  
19 invention is embodied in the system configuration, method of operation, data  
20 format and application code, generally shown in Figures 1 - 7. Application code



1    may be embodied in any form of computer program product. A computer program  
2    product comprises a medium configured to store or transport computer readable  
3    code, or in which computer readable code may be embedded. Some examples of  
4    computer program products are CD-ROM disks, ROM cards, floppy disks,  
5    magnetic tapes, computer hard drives, servers on a network, and carrier waves.

6

7    It will be appreciated that the system, method of operation, data object and  
8    computer product described herein may vary as to the details without departing  
9    from the basic concepts disclosed herein. Moreover, numerous specific details  
10   are set forth in order to provide a more thorough description of the present  
11   invention. However, all specific details may be replaced with generic ones.  
12   Furthermore, well-known features have not been described in detail so as not to  
13   obfuscate the principles expressed herein. While exemplary embodiments of the  
14   present invention described herein is specifically directed to a Macromedia Flash-  
15   XML-MathML based environment, the invention is not limited thereby as one  
16   skilled in the art can readily adapt the concepts presented herein to a preferred  
17   environment. Therefore, other suitable and equivalent programming languages,  
18   platforms and architectures, etc. fall within the scope of the present invention.

19

1 FIG 1 illustrates an exemplary hardware configuration of a processor-controlled  
2 system on which the present invention is implemented. One skilled in the art will  
3 appreciate that the present invention is not limited by the depicted configuration  
4 as the present invention may be implemented on any past, present and future  
5 configuration, including for example, workstation/desktop/laptop/handheld  
6 configurations, client-server configurations, n-tier configurations, distributed  
7 configurations, other networked configurations, etc., having the necessary  
8 components for carrying out the principles expressed herein.  
9

10 FIGS. 1A & B generally depict an advanced versatile layout and rendering system  
11 700 in accordance with the teachings expressed herein, comprising, but not  
12 limited to, a bus 705 that allows for communication among at least one processor  
13 710, at least one memory 715 and at least one storage device 720. The bus 705 is  
14 also coupled to receive inputs from at least one input device 725, e.g., mouse,  
15 keyboard, pen, pad, etc., and provide outputs to at least one output device 730,  
16 monitor, printer, other display medium, etc.. The at least one processor 710 is  
17 configured to perform the techniques provided herein, and more particularly, to  
18 execute the following exemplary computer program product embodiment of the  
19 present invention. Alternatively, the logical functions of the computer program  
20 product embodiment may be distributed among processors connected through

1 networks or other communication means used to couple processors. The  
2 computer program product also executes under various operating systems, such as  
3 versions of Microsoft Windows™, Apple Macintosh™, UNIX, etc.

4  
5 The present invention may be implemented as a computer program product (also  
6 referred to as “QD module”) that is developed for and implemented in the  
7 Macromedia Flash™ environment as, e.g., a Flash™ client application code  
8 module. The QD module in conjunction with a Super-Versatile-Text Display sub-  
9 module (also referred to as “SVT module”) (described below) effectively present  
10 multimedia information on a display output device.

11  
12 FIG. 2 displays an exemplary data format utilized by the QD module in  
13 accordance with the present invention. That the depicted layout and data are  
14 necessarily defined by the environment in which they are used will be apparent to  
15 those skilled in the art. In one embodiment, the QD data format is implemented  
16 as an XML object, an open web standard that is understood by a Flash™  
17 application. The QD data format preferably uses Unicode as the character  
18 encoding, which allows for a huge character set, including most languages and  
19 math symbols. The QD data format also allows for and intermingles styled text,  
20 such as italics, bold, etc., graphics, and mathematical expressions, symbols and

1 other indicia. The graphics may be defined as standard JPEG files or as Flash  
2 SWF™ files and can be animated or interactive. Mathematical expressions are  
3 defined as MathML, an open standard based on XML, which can be imported and  
4 exported by most math software products.

5

6 FIG. 3 depicts an exemplary process flow of the QD module in accordance with  
7 the present invention.

8

9 As shown, at 301, content data, e.g., question data is entered into the system. The  
10 content data includes text, styled text, specifications of external data files  
11 (containing, e.g., graphics or animations), MathML and other displayable objects.  
12 The content data is used to generate main question content and associated answer,  
13 Visual Aid (optionally), and Descriptive Solution (optionally) content.

14

15 At 302, the system processes the question data and converts said data into an  
16 XML tree object. The question data is used to generate text for a main question.

17

18 At 303, the system displays the question text as a Flash™ data block.

19

1 At 304, the system displays potential answer(s) to the main question as a Flash™  
2 data block;

3

4 At 305, the system checks for a Visual Aid related to the main question. As its  
5 name suggests, a Visual Aid, graphical illustrates related question concepts.

6 Depending on the results, processing continues to 306 or 307. If there is a Visual  
7 Aid, processing continues to 306 and then 307. If there is no Visual Aid,  
8 however, processing continues directly to 307.

9

10 At 306, the system displays the Visual Aid as a Flash™ data block and processing  
11 continues to 307.

12

13 At 307 and the system checks for a Descriptive Solution related to the main  
14 question. Depending on the results, processing continues to 308 or 309. If there is  
15 a Descriptive Solution, processing continues to 308 and then 309. If there is no  
16 Descriptive Solution, however, processing continues directly to 309.

17

18 At 308, the system saves the Descriptive Solution for later display as a Flash™  
19 data block and processing continues to 307.

20

1 At 309, the system aligns all data elements according to a desired layout.  
2  
3 At 310, the system displays the question accordingly.  
4  
5 FIG. 7 depicts an exemplary user interface depicting the various elements for  
6 display. As shown, the question text data is presented as Display Area 2, the  
7 potential answer choice(s) data is presented as Display Area 4, the correct answer  
8 data is presented as Display Area 6, the Visual Aid data is presented as Display  
9 Area 8 and the Descriptive Solution data is presented as Display Area 10.  
10  
11 FIG. 4 depicts exemplary pseudo code for implementing the QD module (also  
12 reproduced below).  
13

#### 14 QD Pseudo-code

```
15  
16 function parseQuestionXML  
17     convert raw text to an XML tree  
18     get question layout style from XML  
19 end parseQuestionXML function  
20  
21
```

```
1      function buildQuestionObjects
2          // Sort through branches of question XML.
3          For each branch
4              if the branch is the main question text
5                  create a movieclip to contain the text
6                  call the displaySVTBlock function
7              else if the branch is the answer options
8                  create a movieclip to hold the answers
9                  for each answer
10                     create a movieclip to hold the answer
11                     attach an answer button
12                     create a movieclip to hold the answer text
13                     call the displaySVTBlock function
14                 end for
15             else if the branch is some other content block
16                 if the type of content is visual aid
17                     if this layout calls for a visual aid
18                         create a movieclip to contain the
19 visual aid
20                         call the displaySVTBlock function
21                     else if the type of content is descriptive
22 solution
23                         save the contents for possible later
24 display
25                 end if
```

```
1           end if
2       end for
3   end buildQuestionObjects function
4
5   function layoutQuestion
6       // Positions are based on the question layout style.
7       Position the main question text
8       position the answer block
9       position the answers within the answer block
10      position the visual aid, if required
11      position any other content block
12  end layoutQuestion function
```

13

14

15 FIGS. 5 and 6 depict additional features of the QD module in accordance with the  
16 present invention. Specifically, FIG. 5 illustrates an exemplary process flow of  
17 the Super-Versatile-Text Display module or SVT module. The QD module  
18 interacts with (calls) the SVT module to visually render the QD content data.

19

20 As shown, at 501, content data is entered into the system. This content data is  
21 displayed as follows:

22



1 At 502, the system traverses the XML tree to determine if content (node) is left  
2 to display. If yes, processing continues to 503. If no, processing continues to  
3 505.

4  
5 At 503, the system determines the kind of content left to display. Depending on  
6 the results, the system follows alternate paths. If the content is text, processing  
7 continues to 504A. If the content is an external file, processing continues to  
8 504B. If the content is MathML, processing continues to 504C.

9  
10 At 504A, the system locates a display line that can hold the text data object. The  
11 system then creates a new text object having the appropriate text and style format  
12 and processing returns to 502.

13  
14 At 504B, the system locates a display line that can hold the external file data  
15 object. The system then loads the external file onto the line and processing returns  
16 to 502.

17  
18 At 504C, the system locates a display line that can hold the MathML data object.  
19 The system then renders the MathML object and processing returns to 502.

20

1   When there is no node-data content left to display, processing continues to 505  
2  
3   At 505, the system formats the lines and all data objects within them and displays  
4   the same at 506.  
5  
6   FIG. 6 depicts exemplary pseudo code for implementing the SVT module (also  
7   reproduced below).

8

9           **SVT Display Pseudo-code**

10

11

```
12       // The displayContentBlock function is the interface to  
13       other code.  
14       // External code would call this function, specifying the  
15       xml data to  
16       // display, the destination to display into, and any non-  
17       default  
18       // configuration options.
```

19

```
20       Function displaySVTBlock
```

21

```
1          // Initialize the environment of the destination,
2      based on
3          // configuration options.
4      Set the environment's width
5      set a default text style
6
7      for each node in the XML data
8          if node is text
9              call the displayText function
10         else if node is a visual aid file reference
11             call the loadFile function
12         else if node is MathML
13             call the displayMath function
14         end if
15     end for
16
17     for each line that has been created in destination
18         for each object in line
19             gather measurements
20         end for
21         compute shared baseline and boundaries of line
22         for each object in line
23             position the object so baselines are
24 aligned
25     end for
```

```
1             align line to other lines and destination
2         end for
3
4     end displaySVTBlock
5
6
7
8     function displayText
9
10        inherit the default text style
11        modify the style as specified for this node
12        create an object to hold text within the current line
13
14        while there is text in the node
15            remove a word of text
16            add the word to the current line of destination
17            if current line has exceeded length
18                remove the last line
19                mark the line done
20                create a new current line
21                create an object to hold text within the
22            current line
23            add the word to the current line
24        end if
25    end while
```

```
1
2     end renderText
3
4
5
6     function loadFile
7
8         extract file information from node
9         create an object of the file's given dimensions
10        begin loading the file
11
12        if the object fits in the current line of destination
13            place the object into the line
14        else
15            create a new line
16            if the object doesn't fit into the new empty
17        line
18            scale the object to fit the line
19            end if
20            place the object into the line
21        end if
22
23    end loadFile
24
25
```

```
1
2     function displayMath
3
4         create an object to render the math node into
5         extract MathML data from node
6         call the renderMath function
7
8         if the object fits in the current line of destination
9             place the object into the line
10        else
11            create a new line
12            if the object doesn't fit into the new empty
13        line
14            scale the object to fit the line
15        end if
16        place the object into the line
17    end if
18
19    end displayMath
20
21
22
23    // This function is called recursively - that is, it calls
24    itself.
```

```
1      // MathML objects are frequently composed of other MathML
2      objects,
3      // such as fractions of fractions, so this recursion is
4      necessary.
5      // Nodes in the MathML are of two major types:  composite
6      or terminal.
7      // Composite nodes contain other nodes, while terminal
8      nodes contain
9      // only values, such as a number, variable, or mathematical
10     symbol.
11     // For instance, a fraction node would have two child
12     nodes, the
13     // numerator and denominator.  Each child is rendered
14     separately, then
15     // the first is placed over the other, and a line is drawn
16     between
17     // them.
18
19     Function renderMath
20
21         if the current node is a composite node
22             call the renderMath function on each child node
23             layout the child node based on node type
24         else if the current node is a terminal node
25             if the node contains text
```

```
1             create a text box of the appropriate
2         style
3             else if the node contains an encoded symbol
4                 insert the graphic for that symbol
5             end if
6         end if
7
8     end renderMath
9
```

## 10 EXTERNAL (PUBLIC) FUNCTION DEFINITIONS

11 This section lists the external functions of the Question Display module. While  
12 there is no strict object-oriented public/private status here, these are the only  
13 functions that should be called by outside code. Unless specified, each function  
14 has no return value.

15

### 16 init(initObj)

17 This function should be called once, before calling any of the other functions  
18 below. It initializes the QD environment with various constants, including font  
19 settings and width and height measurements.

### 20 Arguments

21 initObj An object containing any named values to override configuration options

22



1    displayQuestion(question, return\_mc, return\_func)

2    This function displays a question. When display is complete, it calls the specified  
3    return function.

4    Arguments

5    question      The question data, in well-formed XML text.

6    return\_mc      [optional] The context in which return\_func will be called on  
7    completion.

8    return\_func    [optional] The function that will be called within return\_mc.

9

10

11   removeQuestion()

12   This function removes the displayed question.

13   Arguments

14   (none)

15

16   activateAnswers(notify\_mc, notify\_func)

17   This function activates the answer options, making them interactive for the user.

18   When an answer is selected, the specified notification function is called with two  
19   arguments: the letter of the user's selected answer, and the correct answer.

20   Arguments

1    notify\_mc    The context in which notify\_func will be called on completion.

2    notify\_func    The function that will be called within notify\_mc.

3

4    deactivateAnswers()

5    Deactivates all answer options, so that they do not allow user selection.

6    Arguments

7    (none)

8

9    showUserAnswer(userAnswer, showCorrect)

10   This function marks the answer specified in userAnswer, showing whether the

11   selection was correct or incorrect. If showCorrect is set to true and the user's

12   answer was incorrect, the correct answer is also revealed.

13   Arguments

14   userAnswer    The letter of the answer the user has selected.

15   showCorrect    A true/false flag, telling whether to reveal the correct answer.

16

17

18   getCorrectAnswer()

19   Returns the letter of the correct answer for a displayed question.

20   Arguments

1 (none)

2

3 showCorrectAnswer()

4 Reveals to the user the correct answer to a displayed question.

5 Arguments

6 (none)

7

8

9 getAnswerArray()

10 Returns an array of the letters of all the answer options. This is useful for allowing  
11 user selection of an answer via the keyboard.

12 Arguments

13 (none)

14

15 isSolution()

16 This function returns true if there is a descriptive solution available for this  
17 question, and false otherwise.

18 Arguments

19 (none)

20

1    displaySolution(dest\_mc, destWidth, return\_mc, return\_func)

2    This function renders the descriptive solution for the question, if it exists. The  
3    solution is rendered in the specified movieclip, at the specified width. Once the  
4    render is complete, the return function is called.

5    Arguments

6    dest\_mc        The movieclip to render the descriptive solution into.

7    destWidth     The width in pixels of dest\_mc's display area.

8    return\_mc     [optional] The context in which return\_func will be called on  
9    completion.

10   return\_func    [optional] The function that will be called within return\_mc.

11

12   displayXMLBlock(svt\_xml, dest\_mc, destWidth, return\_mc, return\_func)

13   This function renders an XML object of question data (also called an SVT Block)  
14   into the specified movieclip, at the specified width. Once the render is complete,  
15   the return function is called.

16   Arguments

17   svt\_xml        An XML object containing a valid chunk of SVT data.

18   dest\_mc        The movieclip to render the descriptive solution into.

19   destWidth     The width in pixels of dest\_mc's display area.

1    return\_mc     [optional] The context in which return\_func will be called on  
2    completion.

3    return\_func   [optional] The function that will be called within return\_mc.  
4  
5

6    displayTextBlock(svtText, dest\_mc, destWidth, return\_mc, return\_func)

7    Like displayXMLBlock(), this function renders an SVT Block into the specified  
8    movieclip, at the specified width. However, the SVT Block should be passed as  
9    plain text, rather than as an XML object. Once the render is complete, the return  
10   function is called.

11   Arguments

12   svtTextXML   text describing a valid chunk of SVT data.

13   dest\_mc       The movieclip to render the descriptive solution into.

14   destWidth     The width in pixels of dest\_mc's display area.

15   return\_mc     [optional] The context in which return\_func will be called on  
16   completion.

17   return\_func   [optional] The function that will be called within return\_mc.  
18  
19

20   INTERNAL (PRIVATE) FUNCTION DEFINITIONS

1

2 This section lists the internal functions of the QD module. While there is no strict  
3 object-oriented public/private status here, these functions should not be called by  
4 outside code. Any interaction should occur through the External Functions listed  
5 above. Again, unless specified, each function has no return value)

6

7 `parseQuestionXML(rawText)`

8 This function converts the source text for the question into an XML object. It also  
9 checks the XML for the question's layout, which is required before question  
10 rendering can begin. The XML object and layout value are both stored within the  
11 internal question movieclip.

12 Arguments

13 `rawText` A text string containing the well-formed XML for a full question.

14

15 `buildQuestionObjects()`

16 This function sorts through the question XML object, extracting the question text  
17 and answers, as well as any visual aid, descriptive solution, or other content.

18 Movieclips are created for the question text, answers and visual aid, and their

19 SVT blocks are rendered, via the `displaySVT()` function. The descriptive solution,

20 if present, is saved for later display.

1   Arguments

2   (none)

3

4   displaySVT(svt\_xml, svt\_mc)

5   This function takes an SVT Block and renders it into the specified SVT  
6   environment. The SVT Block is an XML object, and the SVT environment is a  
7   movieclip containing the settings and configuration information for SVT display.

8   Arguments

9   svt\_xml       An XML object containing an SVT Block.

10   svt\_mc       The SVT environment movieclip to render the SVT Block into.

11

12

13   getDisplayLine(svt\_mc)

14   This function returns a reference to the current line (a movieclip) in an SVT  
15   environment movieclip. The current line will have at least some room for  
16   additional content (text, graphics, rendered MathML). If the last existing line is  
17   full, or there is no current line, this function will create a new one.

18   Arguments

19   svt\_mc       An SVT environment movieclip.

20

1   endDisplayLine(svt\_mc)

2   This function marks the current display line in the SVT Environment as complete,  
3   so that the next call to getDisplayLine() will return a new line. This is useful for  
4   line breaks, or when a content object must wrap to the next line.

5   Arguments

6   svt\_mc           An SVT environment movieclip.

7

8

9   displayVisualAid(va\_xml, svt\_mc)

10   This function renders a visual aid item into an SVT Environment. The visual aid,  
11   usually a SWF or JPEG, will be loaded from a separate file. The height and width  
12   of the object are specified in the XML, so layout can occur without waiting for the  
13   load to complete. (Loading is accomplished using the piiLoader module.)

14   Arguments

15   va\_xml           An XML node from an SVT Block, containing a visual aid.

16   svt\_mc           An SVT environment movieclip.

17

18   processVisualAid(returnID, va\_mc)



1 This function is called from the piiLoader module when a visual aid object has  
2 completed loading. It completes the processing of the loaded file, verifying that it  
3 fits within the dimensions specified in the XML node of the SVT Block.

4 Arguments

5 returnID A piiLoader Load ID, uniquely identifying this load.

6 va\_mc A movieclip containing the loaded file.

7

8 displayMathML(math\_xml, svt\_mc)

9 This function renders a MathML portion of an SVT Block into a single object.

10 The rendering of individual MathML elements is handled by the renderMathML()

11 function. This function handles the allocation of lines within the SVT

12 Environment, wrapping to the next line if the MathML object is too wide.

13 Arguments

14 math\_xml An XML node from an SVT Block, containing MathML data.

15 svt\_mc An SVT environment movieclip.

16

17 renderMathML(math\_xml, box\_mc, ref\_tf)

18 This function renders individual MathML elements into movieclips containing

19 text and library symbols. The rendering is handled recursively, so that it calls

20 itself to render any MathML elements nested within the main element. (For

1 instance, a the fraction MathML element contains two other elements,  
2 representing numerator and denominator.) This function takes as arguments a  
3 node of MathML data, a movieclip to render that data into, and a text format. The  
4 function returns a reference to the movieclip it creates.

5 Arguments

6 `math_xml` An XML node containing MathML data.  
7 `box_mc` A movieclip to create the new movieclip inside of.  
8 `ref_tf` A text format object, to be used in rendering this object's text.

9  
10 `displayTextItem(rawText, svt_mc)`  
11 This function handles the display of plain and styled text objects from SVT  
12 Blocks. It takes a text string and renders it inside the SVT Environment in the  
13 current text format, splitting the text and wrapping to multiple lines as necessary.

14 Arguments

15 `rawText` A text string.  
16 `svt_mc` An SVT environment movieclip.

17  
18 `drawBorder(a_mc, color, bwidth, bheight)`

19 This function draws a border within a movieclip, using the Flash line-drawing  
20 tools. If `bwidth` and `bheight` are not specified, the measured width and height of

1 the movieclip will be used instead. This function is used extensively within the  
2 rendering functions to force certain measurements onto a movieclip. For instance,  
3 a loaded SWF file might not take up the full space it is meant to occupy,  
4 confusing layout. An invisible border greatly eases such layout computations.  
5 (Note that the line-drawing functions are prone to overwrite any existing lines  
6 within the movieclip.)

#### 7 Arguments

8 **a\_mc**            The movieclip to draw the border into.  
9 **color**            The color to draw the border with. (Only visible while debugging.)  
10 **bwidth [optional]** The width to draw the border.  
11 **bheight[optional]** The height to draw the border.

12

#### 13 layoutQuestion()

14 This function completes the layout of the previously-built question objects. The  
15 question text, answers and visual aid are positioned according to their sizes and  
16 the layout style specified in the question XML.

#### 17 Arguments

18 (none)

19

1 Having now described one or more exemplary embodiments of the invention, it  
2 should be apparent to those skilled in the art that the foregoing is illustrative only  
3 and not limiting, having been presented by way of example only. All the features  
4 disclosed in this specification (including any accompanying claims, abstract, and  
5 drawings) may be replaced by alternative features serving the same purpose, and  
6 equivalents or similar purpose, unless expressly stated otherwise. Therefore,  
7 numerous other embodiments of the modifications thereof are contemplated as  
8 falling within the scope of the present invention as defined by the appended  
9 claims and equivalents thereto.

10

11 Moreover, the techniques presented herein may be implemented in hardware or  
12 software, or a combination of the two. In one embodiment, the techniques are  
13 implemented in computer programs executing on programmable computers that  
14 each include a processor, a storage medium readable by the processor (including  
15 volatile and non-volatile memory and/or storage elements), at least one input  
16 device and one or more output devices. Program code is applied to data entered  
17 using the input device to perform the functions described and to generate output  
18 information. The output information is applied to one or more output devices.

19

1 Each program is preferably implemented in a high level procedural or object  
2 oriented programming language to communicate with a computer system,  
3 however, the programs can be implemented in assembly or machine language, if  
4 desired. In any case, the language may be a compiled or interpreted language. In  
5 one embodiment, the present invention is implemented in the ActionScript  
6 programming language for use in the Macromedia Flash™ environment. The  
7 program code uses Macromedia Flash MX™ to publish, Macromedia Flash  
8 Player™ (e.g., Version 6, Release 48 ,or better) to execute and utilizes the  
9 Macromedia piiiLoader and timeQueue code modules.

10

11

12 Each such computer program is preferably stored on a storage medium or device  
13 (e.g., CD-ROM, NVRAM, ROM, hard disk, magnetic diskette or carrier wave)  
14 that is readable by a general or special purpose programmable computer for  
15 configuring and operating the computer when the storage medium or device is  
16 read by the computer to perform the procedures described in this document. The  
17 system may also be considered to be implemented as a computer-readable storage  
18 medium, configured with a computer program, where the storage medium so  
19 configured causes a computer to operate in a specific and predefined manner.

20

1 The description of the exemplary embodiment herein assumes knowledge of  
2 Macromedia Flash™ and ActionScript™ programming language and a general  
3 understanding of programming documentation conventions. Understanding of  
4 layout and design issues, such as page layout for the web or for print, and  
5 especially as regards the layout of mathematical expressions, will also be useful.

6  
7 Additional aspects and/or features of the present invention include: the code  
8 attempts to provide general solutions as much as possible, but may be specific to  
9 the font and layout size of current implementation. If the font or another part of  
10 the display environment is changed radically, spacing within and between lines  
11 may be tweaked, accordingly.

12  
13 Preferably, math symbols, whether by named entity or Unicode characters, are  
14 usable as follows: both named entities and coded characters are expected to exist  
15 alone within the XML terminal tags. That is, in one embodiment of the present  
16 invention,  $\langle mn \rangle 5 \langle /mn \rangle \langle mn \rangle \pi \langle /mn \rangle$  is valid while  $\langle mn \rangle 5\pi \langle /mn \rangle$  is not.

17

18

1 Finally, an embodiment of the present invention having potential commercial  
2 success is integrated in the Planetii™ Math System™, an online math education  
3 software product, available at <<http://www.planetii.com/home/>>.

4

1 CLAIMS

2

3 What is claimed is:

4

5 1. A display system for multimedia content data comprising

6 Mathematical Markup Language (MathML) data, said system

7 comprising:

8

9 a display medium having a plurality of display lines for rendering

10 multimedia content data thereon;

11

12 a processor associated with said display medium and configured to:

13

14 receive said multimedia content data comprising textual, MathML, and

15 external file indicia data;

16

17 parse said received multimedia content data to derive said textual,

18 MathML and external file indicia data;

19



1           categorize said textual, MathML and file data according to a data  
2           type; wherein said textual data is defined as a TEXT data type, said  
3           MathML data is defined as a MATHML data type and said external  
4           file indicia data is defined as a FILE data type;  
5  
6           store said derived and categorized textual, MathML and external file  
7           indicia data as a tree having a root node and a plurality of offspring  
8           nodes that define left and right subtrees, said root node and said  
9           offspring nodes each having one of said derived textual, MathML and  
10          external file indicia data and respective data type association;  
11  
12          define a traverse procedure that includes:  
13  
14                  visiting a node of said tree,  
15  
16                  determining the data type of said node;  
17  
18                  displaying said node data in accordance with said data type,  
19          wherein:  
20

1           if the data type is TEXT: create a text object having said  
2           textual node data and locate a display line to display said text  
3           object in accordance with predetermined formatting  
4           conventions,  
5  
6           if the data type is FILE: create a file object having said external  
7           file indicia node data and locate a display line to load and  
8           display said file object in-line with previously rendered text  
9           and in accordance with predetermined formatting conventions,  
10  
11          if the data type is MATHML, create a MathML data object  
12          having said stored MathML node data and locate a display line  
13          to display said MathML data object in-line with previously  
14          rendered text and in accordance with predetermined formatting  
15          conventions,  
16  
17          applying said traverse procedure upon the left subtree of said  
18          visited node;  
19

1                   applying said traverse procedure upon the right subtree of said  
2                   visited node; and

3

4                   applying said traverse procedure upon said root node such that  
5                   said root node is the first visited node.

6

7           2. A system as in claim 1 wherein said processor further categorizes said  
8           MathML data as MATHML COMPOSITE and MATHML  
9           TERMINAL data types;

10

11           stores said categorized MathML data as a MathML tree having a root  
12           node and a plurality of offspring nodes that define left and right  
13           subtrees of said MathML tree, said root node and said offspring nodes  
14           of said MathML tree each having one of said derived MathML data  
15           and respective data type association;

16

17           defines a second traverse procedure that includes:

18

19                   visiting a node of said MathML tree,

20

1                   determining the MathML data type of said MathML tree node,  
2  
3                   displaying said MathML tree node data in accordance with said  
4                   data type, wherein:  
5  
6                   if the data type is MATHML COMPOSITE: create a  
7                   MathML data object having said stored MathML  
8                   composite node data and locate a display line to  
9                   display said MathML data object in-line with  
10                  previously rendered text and in accordance with  
11                  predetermined formatting conventions, and,  
12  
13                  if the data type is MATHML TERMINAL: create a  
14                  MathML data object having said stored MathML node  
15                  terminal node data and locate a display line to display  
16                  said MathML data object in-line with previously  
17                  rendered text and in accordance with predetermined  
18                  formatting conventions,  
19

1                   applying said second traverse procedure upon the left sub-  
2                   MathML tree of said visited MathML tree node;  
3  
4                   applying said second traverse procedure upon the right sub-  
5                   MathML tree of said visited MathML tree node; and  
6  
7                   applying said second traverse procedure upon said root node of  
8                   said MathML tree such that said root node is the first visited  
9                   node of the MathML tree.

- 10  
11           3. The system as in claim 1 or 2 wherein said multimedia content data  
12           comprises Markup Language data.  
13  
14           4. The system as in claim 3 wherein said Markup Language data  
15           comprises Extensible Markup Language (XML) data.  
16  
17           5. The system as in claim 1 or 2 wherein said external file indicia data  
18           comprises information associated with data files comprising graphics,  
19           video, animation, other displayable assets or a combination thereof.  
20

1           6. The system as in claim 4 wherein said data files are Macromedia or  
2           Flash-compatible files.

3

4           7. A method of displaying multimedia content data comprising  
5           Mathematical Markup Language (MathML) data, said method  
6           comprising:

7

8           providing a display medium having a plurality of display lines for  
9           rendering multimedia content data thereon;

10

11           receiving said multimedia content data comprising textual, MathML,  
12           and external file indicia data;

13

14           parsing said received multimedia content data to derive said textual,  
15           MathML and external file indicia data;

16

17           categorizing said textual, MathML and file data according to a data  
18           type; wherein said textual data is defined as a TEXT data type, said  
19           MathML data is defined as a MATHML data type and said external  
20           file indicia data is defined as a FILE data type;

1

2

storing said derived and categorized textual, MathML and external

3

file indicia data as a tree having a root node and a plurality of offspring

4

nodes that define left and right subtrees, said root node and said

5

offspring nodes each having one of said derived textual, MathML and

6

external file indicia data and respective data type association;

7

8

defining a traverse procedure that includes:

9

10

visiting a node of said tree,

11

12

determining the data type of said node;

13

14

displaying said node data in accordance with said data type,

15

wherein:

16

17

if the data type is TEXT: create a text object having said

18

textual node data and locate a display line to display said text

19

object in accordance with predetermined formatting

20

conventions,

1

2

if the data type is FILE: create a file object having said external

3

file node data and locate a display line to load and display said

4

file object in-line with previously rendered text and in

5

accordance with predetermined formatting conventions,

6

7

if the data type is MATHML, create a MathML data object

8

having said stored MathML node data and locate a display line

9

to display said MathML data object in-line with previously

10

rendered text and in accordance with predetermined formatting

11

conventions,

12

13

applying said traverse procedure upon the left subtree of said

14

visited node;

15

16

applying said traverse procedure upon the right subtree of said

17

visited node; and

18

19

applying said traverse procedure upon said root node such that

20

said root node is the first visited node.



1

2

8. The method as in claim 7 wherein said method further comprises

3

categorizing said MathML data as MATHML COMPOSITE and

4

MATHML TERMINAL data types;

5

6

storing said categorized MathML data as a MathML tree having a root

7

node and a plurality of offspring nodes that define left and right

8

subtrees of said MathML tree, said root node and said offspring nodes

9

of said MathML tree each having one of said derived MathML data

10

and respective data type association;

11

12

defining a second traverse procedure that includes:

13

14

visiting a node of said MathML tree,

15

16

determining the MathML data type of said MathML tree node,

17

18

displaying said MathML tree node data in accordance with said

19

data type, wherein:

20

1                   if the data type is MATHML COMPOSITE: create a  
2                   MathML data object having said stored MathML  
3                   composite node data and locate a display line to  
4                   display said MathML data object in-line with  
5                   previously rendered text and in accordance with  
6                   predetermined formatting conventions, and,

7

8                   if the data type is MATHML TERMINAL: create a  
9                   MathML data object having said stored MathML node  
10                  terminal node data and locate a display line to display  
11                  said MathML data object in-line with previously  
12                  rendered text and in accordance with predetermined  
13                  formatting conventions,

14

15                  applying said second traverse procedure upon the left sub-  
16                  MathML tree of said visited MathML tree node;

17

18                  applying said second traverse procedure upon the right sub-  
19                  MathML tree of said visited MathML tree node; and

20

1                   applying said second traverse procedure upon said root node of  
2                   said MathML tree such that said root node is the first visited  
3                   node of the MathML tree.

4  
5           9. The method as in claim 7 or 8 wherein said multimedia content data  
6           comprises Markup Language data.

7  
8           10. The method as in claim 9 wherein said Markup Language data  
9           comprises Extensible Markup Language (XML) data.

10  
11          11. The method as in claim 7 or 8 wherein said external file indicia data  
12          comprises information associated with data files comprising graphics,  
13          video, animation, other displayable objects or a combination thereof.

14  
15          12. The method as in claim 11 wherein said data files are Macromedia  
16          Flash or Flash-compatible files.

17

18

19

20

1           13. A user interface for presenting question and answer multimedia data  
2           comprising mathematical MathML content, said system comprising:  
3  
4           a processor configured to:  
5  
6           receive said multimedia data comprising textual, MathML and external  
7           file data;  
8  
9           parse said received multimedia data to derive said textual, MathML  
10          and external file data;  
11  
12          generate question content and associated answer, visual aid and  
13          descriptive solution content based on said derived textual, MathML  
14          and external file data;  
15  
16          display first, second, third and fourth display containers on said  
17          display device, said first display container including a presentation  
18          area for displaying said question content; said second display container  
19          including a presentation area for displaying said answer content; said  
20          third display container including a presentation area for displaying said

1 visual aid content; said fourth display container including a  
2 presentation area for displaying said descriptive solution content; and  
3  
4 wherein for each of said first, second, third and fourth display  
5 containers said MathML and external file data of said container  
6 content are displayed in-line with said textual data of said container  
7 content.

8  
9 14. The user interface as in claim 13 wherein said external file data  
10 includes graphics, video, animation, other displayable objects or any  
11 combination thereof.

12  
13 15. The user interface as in claim 13 or 14 wherein said multimedia data  
14 comprises Markup Language data.

15  
16 16. The user interface as in claim 15 wherein said Markup Language data  
17 comprises Extensible Markup Language (XML) data.

18

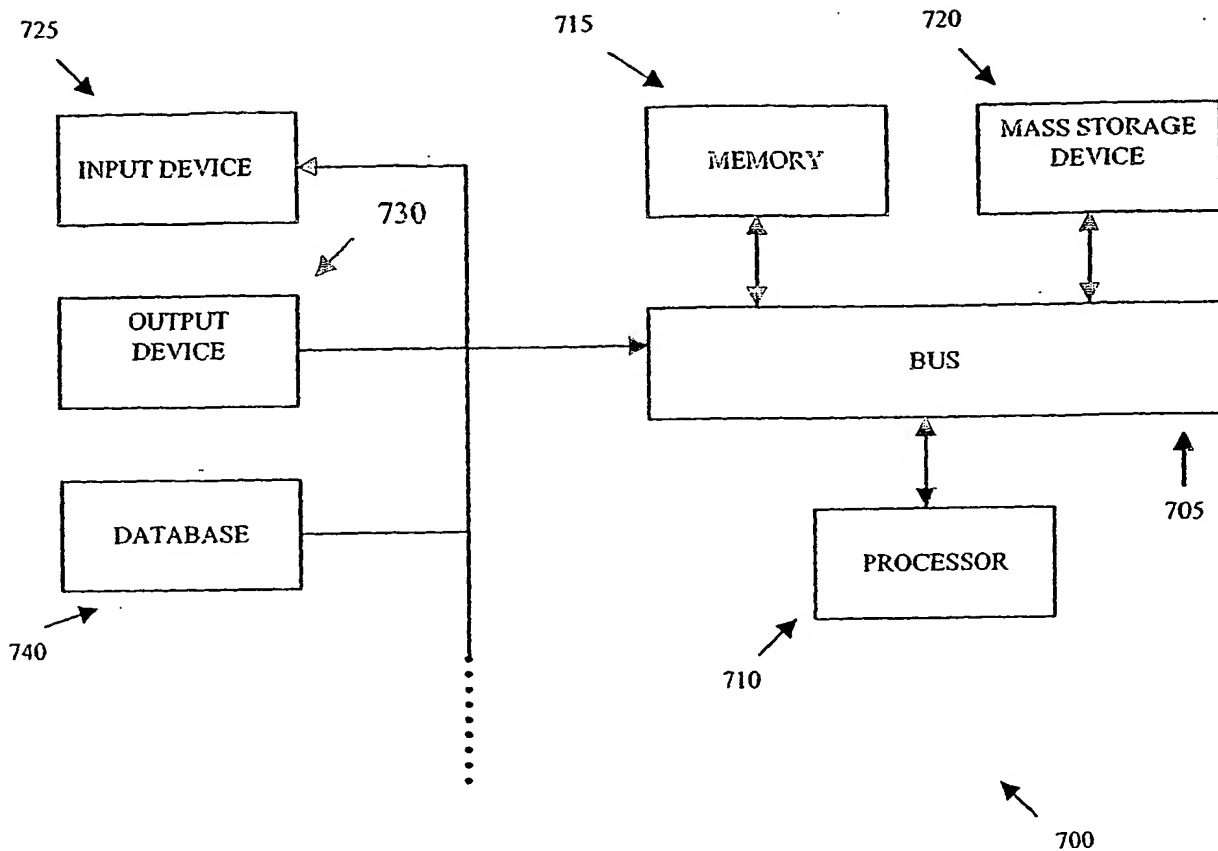


Fig. 1A

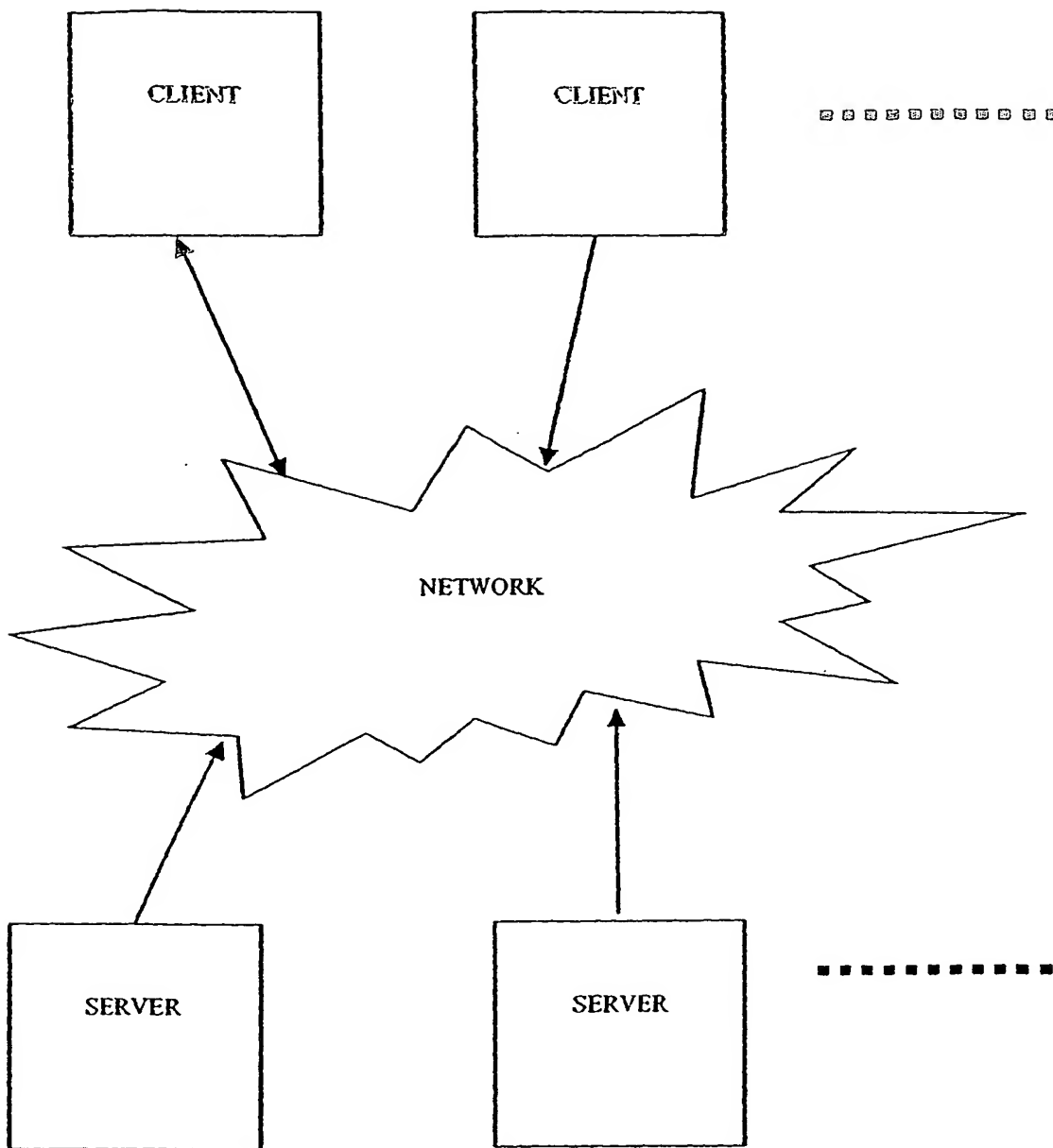


Fig. 1B

**QD Data Structure**

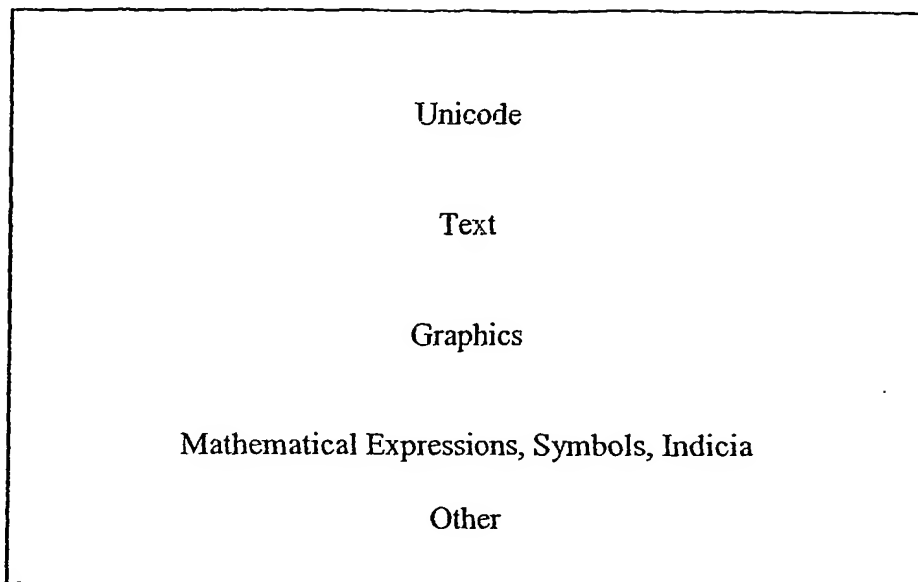


FIG. 2



10/551 661

## Question Display Flowchart

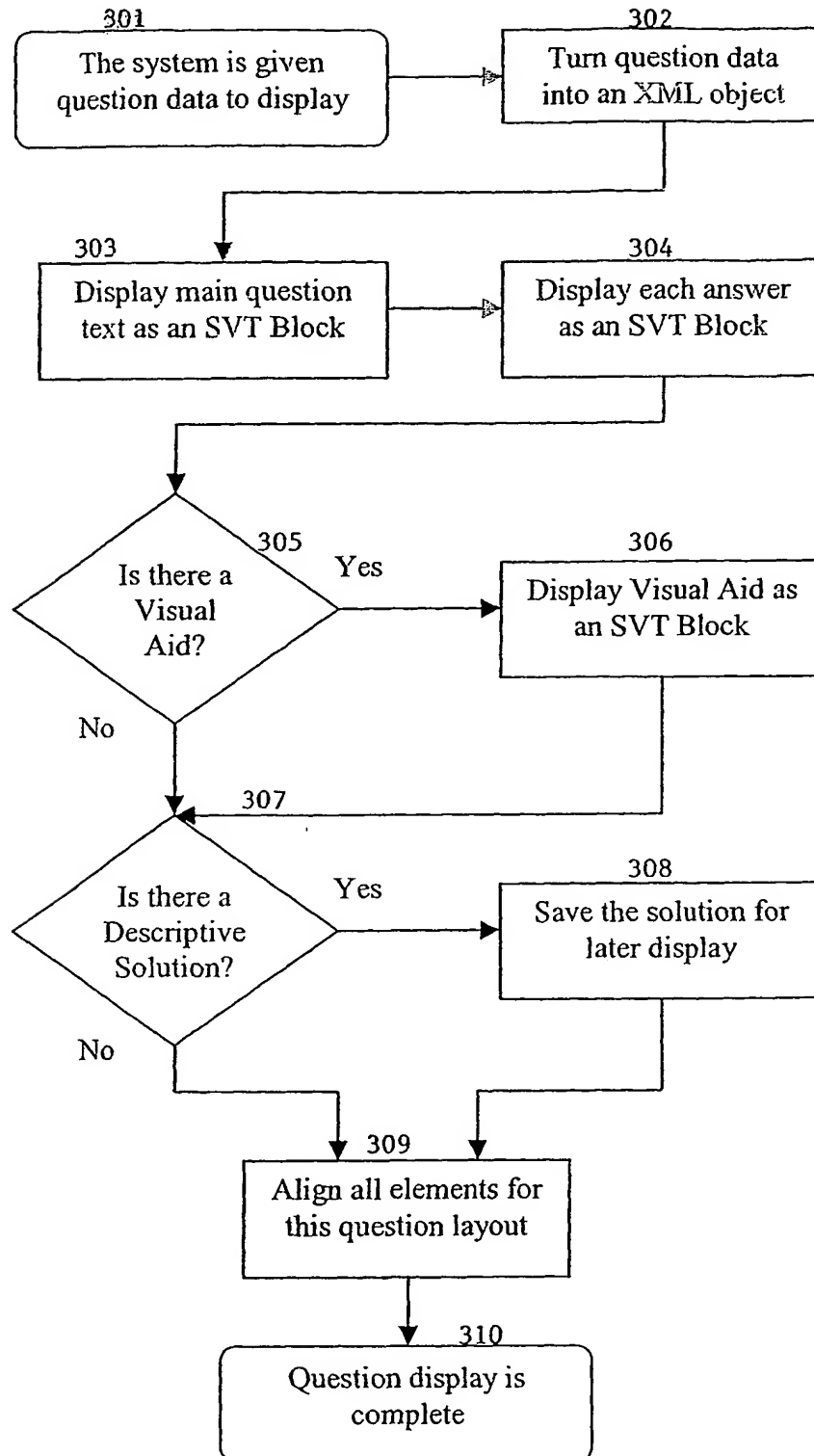


Fig. 3

## Question Display Pseudo-code

```
function parseQuestionXML
    convert raw text to an XML tree
    get question layout style from XML
end parseQuestionXML function
```

```
function buildQuestionObjects
    // Sort through branches of question XML.
    for each branch
        if the branch is the main question text
            create a movieclip to contain the text
            call the displaySVTBlock function
        else if the branch is the answer options
            create a movieclip to hold the answers
            for each answer
                create a movieclip to hold the answer
                attach an answer button
                create a movieclip to hold the answer text
                call the displaySVTBlock function
            end for
        else if the branch is some other content block
            if the type of content is visual aid
                if this layout calls for a visual aid
                    create a movieclip to contain the visual aid
                    call the displaySVTBlock function
                else if the type of content is descriptive solution
                    save the contents for possible later display
                end if
            end if
        end for
    end buildQuestionObjects function
```

```
function layoutQuestion
    // Positions are based on the question layout style.
    position the main question text
    position the answer block
    position the answers within the answer block
    position the visual aid, if required
    position any other content block
end layoutQuestion function
```

Fig. 4

## SVT Display Flowchart

10/551661

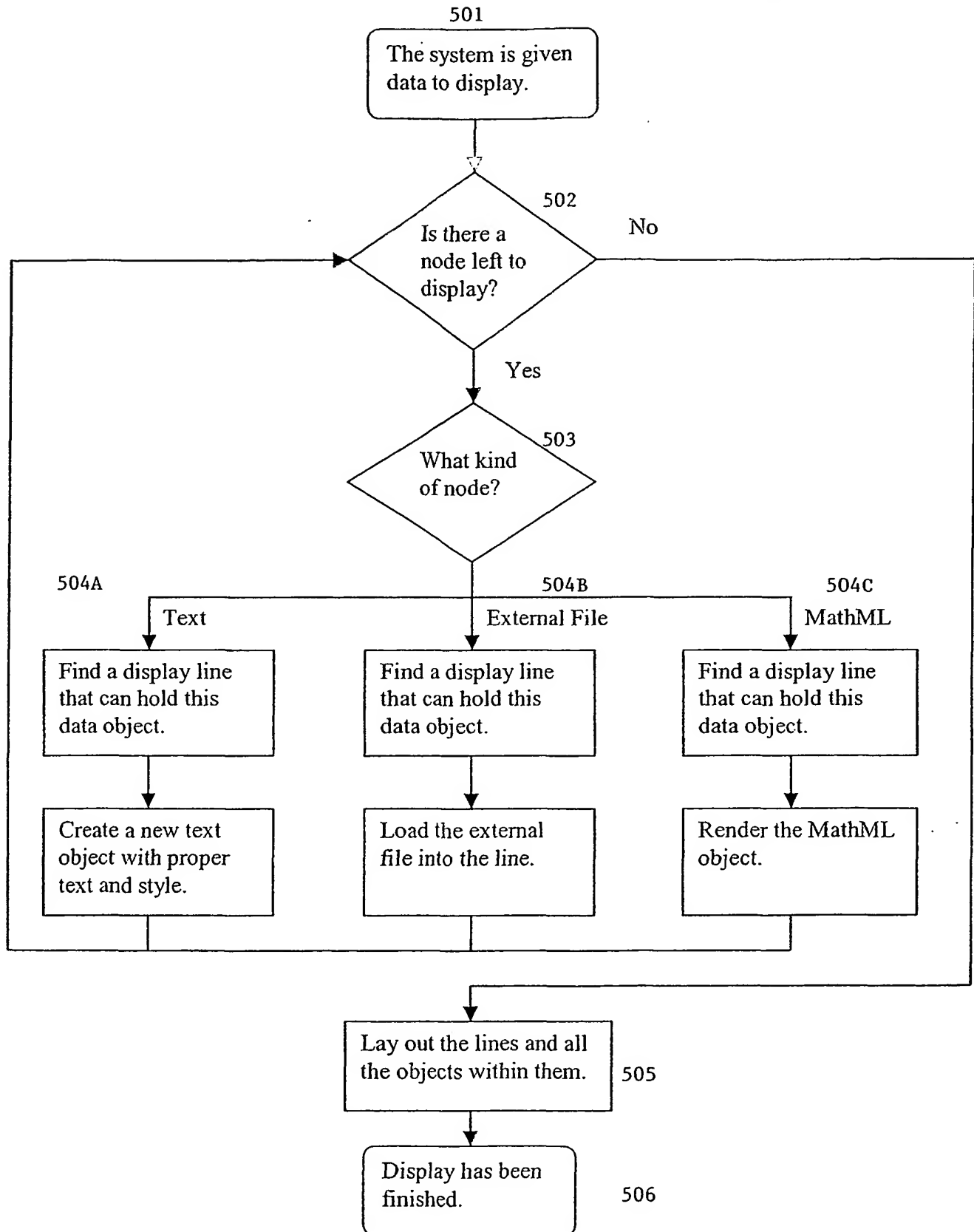


Fig. 5

## SVT Display Pseudo-code

```

// The displayContentBlock function is the interface to other
// code.
// External code would call this function, specifying the xml
// data to
// display, the destination to display into, and any non-default
// configuration options.

function displaySVTBlock
    // Initialize the environment of the destination,
    based on
        // configuration options.
        set the environment's width
        set a default text style

    for each node in the XML data
        if node is text
            call the displayText function
        else if node is a visual aid file
            reference
                call the loadFile function
            else if node is MathML
                call the displayMath function
            end if
        end for

    for each line that has been created in destination
        for each object in line
            gather measurements
        end for
        compute shared baseline and boundaries of
        line
            for each object in line
                position the object so
                baselines are aligned
            end for
            align line to other lines and destination
        end for
    end displaySVTBlock

function displayText
    inherit the default text style
    modify the style as specified for this node
    create an object to hold text within the current line

    while there is text in the node
        remove a word of text
        add the word to the current line of
        destination
            if current line has exceeded length
                remove the last line
                mark the line done
                create a new current line
                create an object to hold text
            within the current line
                add the word to the current
                line
            end if
        end while
    end renderText

function loadFile
    extract file information from node
    create an object of the file's given dimensions
    begin loading the file

    if the object fits in the current line of destination
        place the object into the line
    else
        create a new line
        if the object doesn't fit into the new
        empty line
            scale the object to fit the
            line
        end if
        place the object into the line
    end if
end loadFile

function displayMath
    create an object to render the math node into
    extract MathML data from node
    call the renderMath function

    if the object fits in the current line of destination
        place the object into the line
    else
        create a new line
        if the object doesn't fit into the new
        empty line
            scale the object to fit the
            line
        end if
        place the object into the line
    end if
end displayMath

// This function is called recursively -- that is, it calls
// itself.
// MathML objects are frequently composed of other MathML
// objects.
// such as fractions of fractions. so this recursion is
// necessary.
// Nodes in the MathML are of two major types: composite or
// terminal.
// Composite nodes contain other nodes, while terminal nodes
// contain
// only values, such as a number, variable, or mathematical
// symbol.
// For instance, a fraction node would have two child nodes, the
// numerator and denominator. Each child is rendered separately,
// then
// the first is placed over the other, and a line is drawn
// between
// then.

function renderMath
    if the current node is a composite node
        call the renderMath function on each
        child node
    else if the current node is a terminal node
        if the node contains text
            create a text box of the
            appropriate style
        else if the node contains an encoded
        symbol
            insert the graphic for that
            symbol
        end if
    end if
end renderMath

```

Fig. 6

10

8

6

4

2

Index	PLA/ETII Subtopic Code	Process Step	Sample Question Test					Multiple Choice Answer					Answer
			a.	b.	c.	d.	e.	a.	b.	c.	d.	e.	
70	02N02C	80	354 = tens and 4 ones					5	35	354	30	50	b
71	02N02C	95	There were 85 people in the park. 37 of them were male. How many of them were female?					52 females	42 females	48 females	58 females	122 females	c
72	02N02C	95	Mrs. Kim sold 25 eggs this morning. She sold another 50 eggs in the afternoon. She still has 18 eggs left. How many eggs did she have to begin with?					93 eggs	75 eggs	25 eggs	68 eggs	7 eggs	a
73	02N02C	95	$854 + 354 =$					1,208	1,108	1,158	500	208	a
74	02N02C	100	5 twos =					5 + 5	2 + 2	2 + 2 + 2	55	22	c
75	02N02C	100	4 threes =					4	8	12	6	9	c
78	02N02C	100	Grace is 30 years younger than her father. Her brother is 1 year older than Grace. Grace's father is 57 years old. How old is Grace's brother?					88 years old	27 years old	29 years old	26 years old	28 years old	a

Fig. 7